Parallel Hybrid Particle Swarm Optimization and Applications in Geotechnical Engineering

Youliang Zhang[1], Domenico Gallipoli[2], Charles Augarde[3]

[1] State Key Laboratory for GeoMechanics and Deep Underground Engineering, China University of Mining & Technology , Xuzhou, 221008, P.R. China
ylzhang@whrsm.ac.cn
[2] Department of Civil Engineering, University of Glasgow, Glasgow G12 8LT, UK
gallipoli@civil.gla.ac.uk
[3] School of Engineering, Durham University, South Road, Durham DH1 3LE, UK
Charles.augarde@dur.ac.uk

Abstract. A novel parallel hybrid particle swarm optimization algorithm named hmPSO is presented. The new algorithm combines particle swarm optimization with a local search method which aims to accelerate the rate of convergence. The hybrid global optimization algorithm adjusts its searching space through the local search results. Parallelization is based on the client-server model, which is ideal for asynchronous distributed computations. The server is the center of data exchange, which manages requests and coordinates the time-consuming computations undertaken by individual clients. A case study in geotechnical engineering demonstrates the effectiveness and efficiency of the proposed algorithm.

Keywords: particle swarm optimization; asynchronous parallel computation; server-client model; hmPSO.

1 Introduction

Global optimization algorithms are important tools for parameter identification in geotechnical engineering. However, due to the lack of analytical solutions for most geotechnical problems, evaluation of the objective function $f(\mathbf{x})$ during optimization is usually achieved by performing numerical (e.g. finite element) simulations of the relevant boundary value problem. This approach, which is often referred to as "simulation-based optimization" might produce multi-modal forms of the objective function due to numerical fluctuations and is generally computationally expensive.

Particle Swarm Optimization (PSO) proposed by Eberhart and Kennedy [1, 2] is a new evolutionary algorithm inspired by social behavior of bird flocks. Recently, PSO has been receiving increasing attention due to its effectiveness, efficiency, and simplicity in achieving global optimization. In particular, PSO is capable of capturing the global optimum for a wide range of multi-modal problems (i.e. those problems where the objective function might have many local minima). However, for some complex problems, PSO could suffer from premature convergence towards a

suboptimal solution, just like other evolutionary algorithms. Another weakness of PSO is the slow rate of convergence, especially in the later stage of the search [3, 4].

A modified parallel PSO is here proposed to address shortfalls such as premature identification of suboptimal solutions, slow convergence rate and high computational costs. In particular, a hybridization technique is introduced where a local search is run in parallel with the main PSO algorithm to carry out quick and efficient explorations around the current optimum at a much lower computational cost. The search space for the PSO algorithm is centered on the latest solution from the local search routine and is progressively narrowed as the algorithm progresses. Hybridization of a global optimization method such as PSO with a local search method has proved to be very effective in solving a range of problems [5-7]. In addition, an asynchronous parallel version of the hybrid PSO algorithm is here proposed to reduce computational time. The main parts of the method were presented in a previous paper [8]. The implementation details and an application to a typical geotechnical problem are instead the focus of this paper.

2 Parallel hybrid particle swarm optimization

PSO is advantageous for global exploration of the search space while local search methods offer fast convergence rates when good starting points are provided. The hybrid PSO algorithm presented here aims to combine the strengths from both these approaches. The proposed parallel hybrid PSO algorithm consists of the basic PSO, a local search method and an asynchronous parallel strategy. The Message Passing Interface (MPI) [9] is chosen as the parallel programming library.

2.1 Basic PSO

The PSO algorithm was proposed as a population-based stochastic global optimization method by Kennedy & Eberhart [1, 2]. The population, which is called "swarm" in PSO, is made up of "particles". The ith particle has such properties as fitness f_i, position $\mathbf{x_i}$, and velocity $\mathbf{v_i}$. The PSO starts by initializing the particle positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ and computes the corresponding fitness f_i. Then particles "fly" across the search space, which means that a series of iteration is performed where, for each iteration, the fitness of all particles is updated according to the following rules

$$\mathbf{v}_i^{k+1} = w^k \mathbf{v}_i^k + c_1 r_1 \left(\mathbf{P}_i - \mathbf{x}_i\right) + c_2 r_2 \left(\mathbf{P}_g - \mathbf{x}_i\right). \quad \textbf{(1)}$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1}. \quad \textbf{(2)}$$

where the superscripts k, $k+1$ are iteration numbers; r_1 and r_2 are two random factors in the interval [0,1]; w_k is the inertia weight and c_1 and c_2 are constants representing the "cognitive" and "social" components of knowledge, respectively. Each particle

remembers its best position (cognitive knowledge), which is denoted as $\mathbf{P}_i$. In addition, knowledge of the best position ever achieved across the whole swarm, which is denoted as $\mathbf{P}_g$, is shared between particles (social knowledge). Updating rules drive particles towards the optimal region and, as the algorithm progresses, all particles tend to group around the global optimum until a solution is finally found.

2.2 Local search method

Local search methods are a class of methods that hunt for the optimum solution in the vicinity of a given set of starting points. The particular choice of these starting points strongly affects the efficiency of the local search. The Nelder-Mead simplex algorithm [10] is used herein mainly due to its simplicity and conformity with PSO. Both Nelder-Mead and PSO methods require only evaluation of the objective function at given points in the search space and no gradient information is needed, which makes the two methods easy to combine. The simplex method needs $n + 1$ points as starting points, where n is the number of unknown parameters to be determined.

2.3 Asynchronous parallel hybrid PSO

The objective of the hybrid method described here is to make best use of the strengths of the global and the local search methods. PSO is powerful for global exploration of the search space while the Nelder-Mead simplex algorithm is efficient for local exploration. The combination of the two methods is achieved by taking the n+1 best solutions from the PSO algorithm as starting points of the Nelder-Mead simplex algorithm. The solution from the local search is then fed back into the PSO as a candidate global optimum. At the same time, a sub-swarm is allocated to explore a smaller region centered on the solution obtained from the local search. In the proposed algorithm, which is named hmPSO (hybrid moving-boundary Particle Swarm Optimization), the whole swarm is therefore divided into two sub-swarms. The first “global” sub-swarm searches in the original space while the second “local” sub-swarm searches in a smaller space that is continuously updated to coincide with the region around the latest solution from the local search.

In the parallel implementation of the hmPSO algorithm each particle is assigned to a different processor. Parallelization can be achieved by using either a synchronous or asynchronous, model. In the synchronous model, PSO moves to the next iteration only when all particles have completed their current iteration, i.e. when all particles have terminated the numerical simulation corresponding to their current position in the search space. A drawback of the synchronous model is that the slowest particle determines the speed of the algorithm and, in non-linear simulation-based optimization, the computational time can vary significantly between particles. In contrast, in the asynchronous model, each particle moves to the next iteration immediately after it finishes the current one without having to wait for other particles. This feature is particularly advantageous when load unbalances between particles are very large as no processor remains idle while other processors are busy. The use of the asynchronous model is also essential in distributed parallel computing where

computing power of different processors vary significantly due to hardware configurations.

The client-server model is adopted for the implementation of the asynchronous model as shown in Fig. 1. The client-server model consists of three parts: the server, the particle clients and the local search clients. The server resides on one processor, which is the centre for data sharing and system management. The server stores and sorts the best positions of individual particles and updates the swarm's best position. The clients are instead responsible for undertaking the actual numerical simulations corresponding to the current position of their respective particles and for running the Nelder-Mead simplex local searches. Each client communicates with the server, and there is no communication between particle clients. There is only one local search client when a serial local search algorithm is used. Otherwise, if a parallel local search algorithm is employed, there is more than one local search client. In this case a master-slave model is adopted for the parallel local search where the master of the local search group coordinates and manages computations undertaken by the slaves. When a master-slave model is adopted, the server only communicates with the master of the local search group. The number of clients depends on the dimension of the problem and the specific algorithm used.

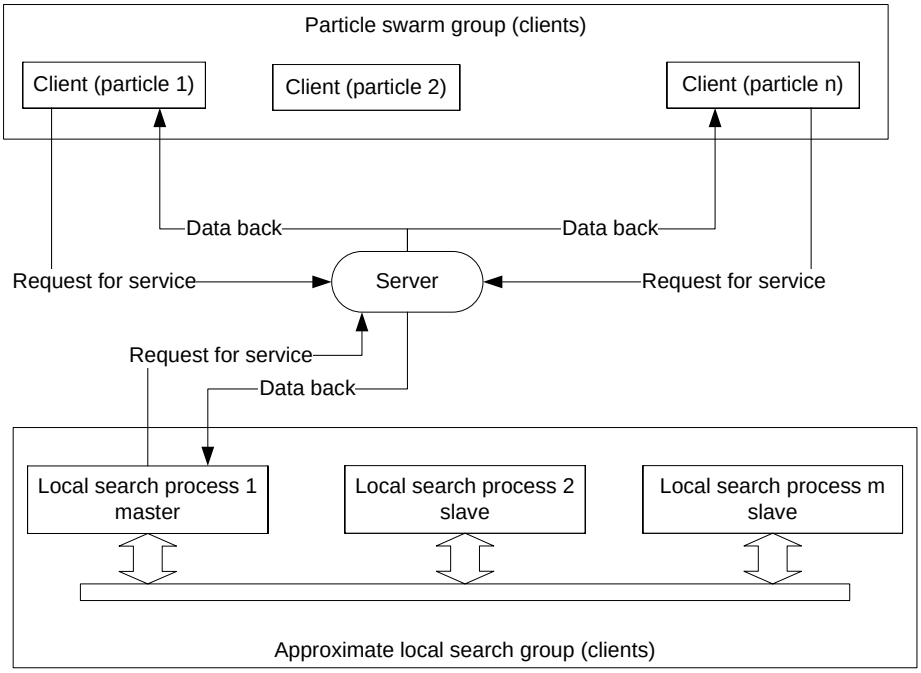

Fig. 1. Client-server model for parallel hmPSO

The client's responsibility is to perform the actual numerical simulations for evaluating the objective function and to undertake the local searches. The server's responsibility is to store and manage data shared by clients, listen to clients' queries for information or data, process the queries and return data or commands back to the

clients. The server is also in charge of termination of all clients on receipt of stop information either from the local search client or from a particle client. It is obvious that different processors run different programs using different data, so the paradigm of the parallel hmPSO is MPMD (Multiple Programs Multiple Data).

3 Parallel implementation of hmPSO

Non-blocking MPI communication functions such as MPI_Isend, MPI_Irecv, and MPI_Test are used for the implementation of the asynchronous parallel hmPSO. By using non-blocking communication which allows both communications and computations to proceed concurrently, the performance of the parallel program can be improved. For example, MPI_Isend starts a send operation then returns immediately without waiting for the communication to complete. Similarly, MPI_Irecv starts a receive operation and returns before the data has been received. The structure for a communication overlapping with a computation is,

MPI_Isend
Do some computation
MPI_Test

Checking for the completion of non-blocking send and receive operations is carried out by MPI_Test. This check should be performed before any data being exchanged between processors is used. MPI_Test waits until a non-blocking communication has finished but it does not block the algorithm so to avoid any deadlock. More details on how to use these functions can be found in [9].

The three parts of the algorithm, i.e. the server program, the particle client program, and the Nelder-Mead simplex program, are listed below. In the parallel algorithm, each processor hosts only one client.

```
parallel hmPSO  (program resides on the server)
  repeat
  MPI_Irecv (receiving data from all clients )
  MPI_Test
  switch according to client data/request
    if ( data is particle best solution )
      update swarm best
      send swarm best back to the particle
      If (FES>startFES)
      send starting points to clients of local search
    else if ( data is local search solution )
      update swarm best
      sort particles by their best fitness
      send starting points to local search
      restart a sub-swarm search on a smaller search
space based on the local search solution
    else if ( data is stop message )
      send stop message to all clients
      terminate server
    end if
    check stopping criterion
    if (stopping criterion is met )
```

```
      send stop message to all clients
      terminate server
    end if
  }while ( stopping criterion is not met )
```

The server program keeps on receiving data or requests from all clients, and responds accordingly. The server also stores the best position of the whole swarm and the best positions of individual particles. The swarm best position is sent back to particle clients when a request is made.

After sorting the best positions of individual particles, the top n+1 particles are taken as the starting points for the Nelder-Mead simplex local search. After termination of the Nelder-Mead simplex local search, a sub-swarm is allocated to explore a smaller search space centered on the solution from the local search. The server is also in charge of the termination of all clients by sending a stop message.

```
parallel hmPSO  (program resides on particle clients)
  initialize position x_i
  initialize velocity v_i
  evaluate objective function f(x_i)
  update particle best P_i
  MPI_Isend (send particle best P_i to server)
  repeat
    MPI_Irecv (receive data/command from server )
    MPI_Test
    update velocity v_i
    update position x_i
    evaluate objective function f(x_i)
    switch according to the received data from service
      if ( data is stop message )
        terminate this client process
      else if ( data is swarm best position P_g )
        update P_g for this particle
      else if ( data is to restart sub-swarm)
        reset particle's new search space
        initialize position x_i
        initialize velocity v_i
      end if
    check stopping criterion
    if (stopping criterion is met )
      send stop message to the server
    else
      MPI_Isend (send particle best P_i to server )
    end if
  }while ( stopping criterion is not met )
```

Particle clients update their respective positions and velocity by performing the relevant numerical simulation and evaluate the objective function. These are the most computationally expensive operations of the whole optimization process. The particles receive the swarm best position $\mathbf{P}_g$ from the server and, after completing the numerical simulation, return the particle best position $\mathbf{P}_i$ to the server. When the convergence criterion is met, the clients are commanded by the server to terminate their processes.

```
parallel hmPSO  (program resides on local search client)
  MPI_Irecv (receive data/command from server )
  MPI_Test
  repeat
    switch according to the received data from service
      if ( data is stop message )
        terminate this client process
      else if ( data is on starting points )
        perform a local search
      end if
    check stopping criterion
    if (stopping criterion is met )
      send stop message to the server
    else
      MPI_Isend (send local search solution to server )
    end if
  }while ( stopping criterion is not met )
```

The local search client starts a new run of the Nelder-Mead simplex algorithm as soon as it receives a fresh set of n+1 starting points from the server. The solution from the local search is then sent back to the server to update the current best position of the whole swarm. When the convergence criterion is met, the local search client sends a message to the server.

4 Applications to geotechnical engineering

The above parallel hmPSO algorithm is here used for selecting parameter values in the Barcelona Basic Model (BBM) [11] based on the results from pressuremeter tests. BBM .is one of the best known constitutive models for unsaturated soils and is formulated in terms of 9 independent parameters. In this work, however, only a sub-set of 6 parameter values is determined. This follows a sensitivity analysis that has shown the dominant influence of these 6 parameters in governing the soil response during simulations of pressuremeter tests.

Pressuremeter testing is a widely used in-situ technique for characterizing soil properties. The technique consists in the application of an increasing pressure to the walls of a cylindrical borehole while measuring the corresponding radial strains. Cavity expansion curves showing applied pressure versus radial expansion are then analyzed to infer soil properties. Due to the nonlinearity of BBM, no closed-form analytical solution exists to predict soil behaviour during a pressuremeter test and, hence finite element simulations are here used.

In order to validate the proposed algorithm, three simulated constant suction pressuremeter tests corresponding to a known set of parameter values in BBM (see Fig. 2) were taken as the “experimental” data. The optimization algorithm was then tested to check whether the same set of parameter values could be found. The objective function is given in Eq. 3 and is defined in such a way that the three pressuremeter tests at different suctions have to be simultaneously matched

$$f = \sqrt{\sum_{i=1}^{N} \left(\varepsilon_{c,i}^{s} - \varepsilon_{c,i}^{m} \right)^2} \quad . \tag{3}$$

where ($\varepsilon^{s}_{c,i}$ - $\varepsilon^{m}_{c,i}$) is the difference between the "experimental" and simulated cavity strains at the same value of cavity pressure, and N is the total number of "experimental" points on the three curves.

The range of parameter values defining the entire 6-dimensional search space are given in Table 1. The optimum solution in this search space is **x**=(M, k, κ, r, β, p^c)=(0.9, 0.5, 0.025, 1.5, 1.0e-5Pa^{-1}, 2.0e6Pa), corresponding to the set of BBM parameter values used to generate the curves shown in Fig. 2. The other three BBM parameters that were not included in the optimization process were taken equal to (G, λ(0), po*)=(3.0e+3 kPa, 0.13, 9.18 kPa) in all simulations. Readers can refer to [11] for the physical meaning of the individual model parameters.

Pressuremeter tests were simulated by a 2D axisymmetric FE model using eight-noded quadrilateral elements with pore water pressure and pore air pressure on the corner nodes. The cavity pressure was applied incrementally in steps of 10 kPa.

In the PSO algorithm, the swarm size was 35, the constants c_1 and c_2 were both taken equal to 2.0, the convergence tolerance for the objective function was equal to 1.0e-5 and the inertia weight w_k decreased linearly with the number of iterations from 0.9 to 0.4.

The parallel computer "Hamilton" was used for the analysis. This is a Linux cluster hosted at Durham University consisting of 96 dual-processor dual-core Opteron 2.2 GHz nodes with 8 GBytes of memory and a Myrinet fast interconnect for running MPI code, and 135 dual-processor Opteron 2 GHz nodes with 4 GBytes of memory and a Gigabit interconnect. The system has 3.5 Terabytes of disk storage. The operating system is SuSE Linux 10.0 (64-bit).

Table 1. BBM parameters and their ranges.

parameter	Minimum value	Maximum value
M	0.1	1.4
k	0.1	0.7
κ	0.01	0.1
r	1.05	1.8
β	1.0e-6 kPa^{-1}	1.0e-4 kPa^{-1}
p^c	1.0e4 kPa	1.0e7 kPa

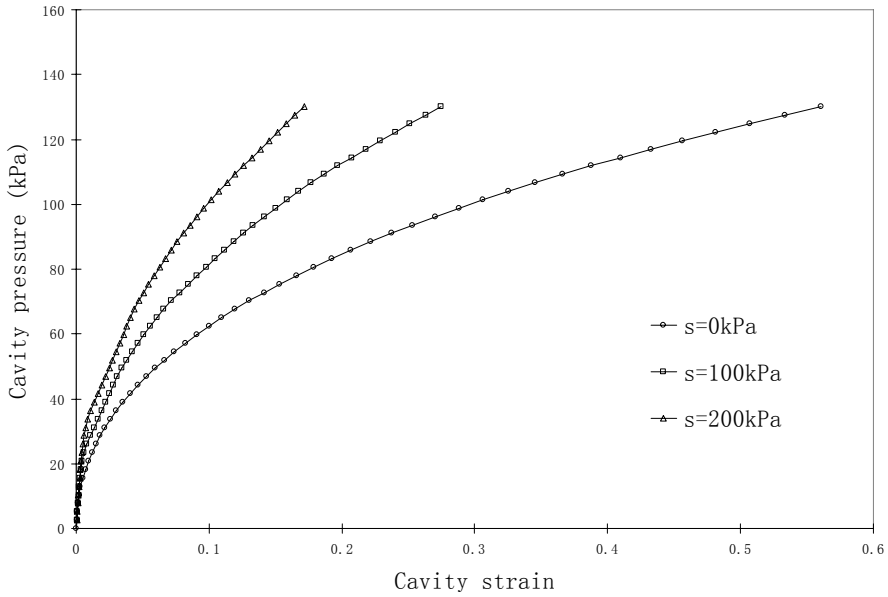

Fig. 2. Curves generated by the chosen parameter set

Results show that the algorithm was capable to find the target 6 parameter values with very high accuracy. The optimum parameter values were found at the end of a local search and it took 4 local searches to converge to the solution. As an example, the first 7 rows in Table 2 show the initial values at the 7 corners of the simplex for the last local search while the bottom row shows the final optimum solution. The hmPSO algorithm took a total number of 34884 objective function evaluations to converge towards the solution. This corresponded to a computational time of 191.6 hours for the asynchronous parallel implementation. The average time of a single objective function evaluation is 122.2 seconds. So it can be estimated that if a serial algorithm is used, the computation time could be over 1184.2 hours. This confirms the advantage of using a parallel implementation rather than a serial one.

Table 2. Results from the last local search.

objective function value	M	k	κ	r	β (kPa^{-1})	p_c (kPa)
2.71E-05	9.00E-01	5.00E-01	2.50E-02	1.50E+00	9.99E-06	2.01E+06
3.04E-03	9.14E-01	4.54E-01	2.12E-02	1.32E+00	1.14E-05	1.62E+06
3.22E-03	9.14E-01	4.56E-01	2.14E-02	1.33E+00	1.17E-05	1.59E+06
3.24E-03	9.13E-01	4.53E-01	2.16E-02	1.32E+00	1.13E-05	1.61E+06
3.38E-03	9.11E-01	4.54E-01	2.12E-02	1.33E+00	1.12E-05	1.90E+06
3.41E-03	9.17E-01	4.54E-01	2.18E-02	1.32E+00	1.12E-05	1.30E+06
3.54E-03	9.12E-01	4.54E-01	2.18E-02	1.33E+00	1.14E-05	1.68E+06
2.0618E-06	9.00E-01	5.00E-01	2.50E-02	1.50E+00	1.00E-05	2.00E+06

5 Conclusions

The paper presents the implementation and application of a parallel hybrid moving-boundary Particle Swarm Optimization algorithm (hmPSO). The algorithm originates from the hybridization of the basic particle swarm optimization algorithm with a Nelder-Mead simplex local search. A client-server model is used for the asynchronous parallel implementation of the algorithm. Using the proposed methodology, 6 parameter values of a nonlinear constitutive unsaturated soil model were simultaneously identified by means of back analysis of pressuremeter tests. Computational time was reduced significantly by parallelization of the algorithm on the computer cluster "Hamilton" at Durham University, UK.

Acknowledgements
The authors gratefully acknowledge support from U.K. EPSRC (grant ref. EP/C526627/1) and State Key Laboratory for GeoMechanics and Deep Underground Engineering (grant SKLGDUE08003X).

References

1. Kennedy J. and Eberhart R. Particle swarm optimization. In: IEEE, NeuralNetworks Council Staff, IEEE Neural Networks Council editor. Proc. IEEE International Conference on Neural Networks. IEEE.1942–1948 (1995).
2. Eberhart R. and Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya Japan, 39–43 (1995).
3. Xie X., Zhang W. and Yang Z. A dissipative particle swarm optimization. in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), Hawaii, USA,1456–1461(2002**)**.
4. Zhang W., Liu M. and Clerc Y. An adaptive pso algorithm for reactive power optimization. In: Sixth international conference on advances in power system control, operation and management (APSCOM) Hong Kong, China, 302-307 (2003).
5. Renders J. and Flasse S. Hybrid methods using genetic algorithms for global optimization. IEEE Trans Syst Man Cybern B Cybern. 26(2), 243–258 (1996).
6. Yen R., Liao J., Lee B. and Randolph D. A hybrid approach to modeling metabolic systems using a genetic algorithm and Simplex method. IEEE Transactions on Systems, Man and Cybernetics Part-B, 28(2),173–191 (1998).
7. Fan S., Liang Y. and Zahara E. Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions. Engineering Optimization. 36, 401–418 (2004).
8. Zhang Y., Gallipoli D. and Augarde C.E. Simulation-based calibration of geotechnical parameters using parallel hybrid moving boundary particle swarm optimization. Computers and Geotechnics. 36 (4), 604-615 (2009).
9. Snir M., Otto S., Huss-Lederman S., Walker D. and Dongarra J. MPI: The Complete Reference. MIT Press (1996).
10. Nelder J. and Mead R. A simplex method for function minimization. The Computer Journal. 7, 308–313 (1965).
11. Alonso E.E., Gens A. and Josa A. A constitutive model for partially saturated soils. Géotechnique. 40(3), 405-430 (1990).